Seoul inter-noise 2020 23-26 AUGUST

Current-saving sampling for the embedded implementation of positive position feedback

Gergely Takács[1,⋆], Erik Mikuláš[2], Martin Vríčan[3] and Martin Gulan[4]
Slovak University of Technology in Bratislava
Faculty of Mechanical Engineering
Institute of Automation, Measurement and Applied Informatics
Námestie slobody 17, 812 31 Bratislava, Slovakia

ABSTRACT

Battery powered embedded applications of active vibration control must conserve energy as much as possible, in order to prolongate effective service time between charging. Besides the power requirements of actuators and sensors, the microcontroller unit executing the feedback control algorithm drains the battery as well. Reducing the current drain of the microcontroller is possible by slowing down its clock frequency; or, sending the device into sleep mode when idle. This paper attempts to provide a definitive answer as to which of these strategies is better. A simple theoretical approximation derived here suggests that running the microcontroller at maximum frequency, then sending it to a low power mode is always preferable. We confirm this by laboratory experiments measuring the microcontroller power requirements of the positive position feedback algorithm. For this algorithm and choice of sampling time a five-fold current saving is possible on the Microchip (Atmel) ATmega 328p microcontroller unit, reducing its current draw from 15 mA to 3 mA. The experimental current and instantaneous power measurements shown in the paper confirm our prediction and general conclusion.

1. INTRODUCTION

Active vibration control (AVC) has now left the realm of basic research and became mainstream technology in numerous consumer and professional application areas. Certain AVC implementations are now even battery-powered, such as active utensils helping patients suffering from tremors [5], hand-operated power tools with vibration suppression [15] or optical stabilization systems and gimbals [2].

Such battery powered embedded systems must minimize their power consumption to conserve battery life. The working voltage of microcontroller units (MCUs) and their cores are constantly decreasing for this very reason. Certain modern MCUs even offer dynamic voltage and frequency scheduling [10], in addition to the general trend of declining logic voltage levels. Besides hardware changes, appropriate embedded software design practices greatly influence energy requirements.

[1]gergely.takacs@stuba.sk, ⋆ corresponding author

[2]erik.mikulas@stuba.sk

[3]vrican.mt@gmail.com

[4]martin.gulan@stuba.sk

The second major factor contributing to power requirements besides rail voltage is microcontroller current, which linearly depends on clock speed and it is a parameter commonly listed in the units of μA/MHz [3]. Unfortunately, microcontrollers consume approximately the same current even if they are not performing any useful task, since the transistors making up the chip must be switched anyways. Thus, portable devices spend most of their time in so-called low power modes (LPM) [8]. Although terminology differs among architectures and manufacturers; *sleep*, *standby*, *hibernation*, *shutdown* or *power down* regimes refer to a certain level of dramatically reduced current consumption, coming at the price of preserving only essential microcontroller functionality. Hence, the time spent in active mode plays a major role in energy consumption, with average current simply expressed as a ratio of time spent in active and sleeping modes [3,17]. While active current consumption is mainly influenced by clock speed, LPM drain is composed of transistor leakage and the consumption of peripherals enabling waking from the LPM, such as the watchdog timer (WDT) or brownout detection (BOD) circuitry [17]. A 80-90% power reduction is common, but there are microcontrollers on the market consuming only 20 nA in sleep [17].

The effect of frequency throttling and low power modes on the analog-to-digital converter (ADC) peripheral are investigated in [14], while Zuo and Liu review general power reduction strategies in [17]. A complete characterization of the MSP430 microcontroller providing an exhaustive clock frequency vs. current consumption is given in [3]. The relationship of energy efficiency and embedded software has been investigated in [1], where the effect of elementary mathematical operations on power consumption, including serial communication bitrate and other factors is taken into account. Although both theoretical and experimental models of instruction-level power consumption are available (cf. [1, 4]), the most important factors are still time spent in active mode and the clock frequency used.

Developing a strategy to minimize power consumption of an arrythmia detecting implant is discussed in [3], while Launtner et al. presented a more exhaustive algorithm to schedule resources for real-time tasks utilizing the lowest possible MCU sleep modes with extended wake-up times [10]. The problem of time-resource allocation received extensive attention in literature (cf. [9, 10]), although this problem is more linked to real-time operating systems that must run several pseudo-simultaneous and even nested tasks with different priorities, execution times and periodic time constraints.

In this paper, we consider the specific class of embedded applications running feedback control algorithms, such as those needed for active vibration control. In this context, the advanced results of resource allocation and scheduling are not directly applicable, as we have a single computation to be completed with hard real-time constraints. Protracted idle modes are not possible to implement, one has to decide between throttling clock speed to fill sampling times at lower power levels or putting the devices into sleep mode at idle times and waking it up at each sample. Our goal is then to provide a viable generic recommendation, answering: 'Which strategy is most suitable for control systems with sub-second periodic sampling and certain task execution times (TET), such as positive position feedback (PPF) in active vibration control?' In the following sections we attempt to derive a general model and verify its validity through experimental measurement on a miniature flexible-link laboratory device.

2. CURRENT-SAVING FOR DIGITAL FEEDBACK CONTROL

Control algorithms must be evaluated with hard real-time constraints, meaning that the processing of the input measurement, control decision and, finally, actuation takes less time than the chosen sampling period T [s]. Figure 1 illustrates the timing of control algorithms implemented in digital computers.

As the total execution time (TET) t [s] must be always $t < T$, the rest of the time the microcontroller has no computations to evaluate. From the viewpoint of energy efficiency, this time is wasted, since

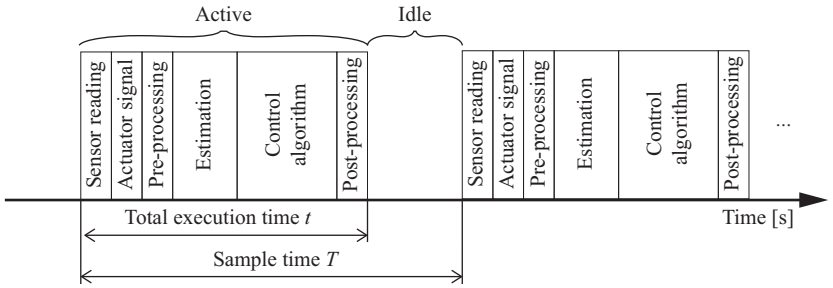


Figure 1: Timing of control algorithms.

the current consumption of microcontrollers directly depends on clock speed. No matter if there are no actual computations performed, the fundamental transistors making up the chip must still be switched at every clock cycle.

This problem is of course well known, and embedded system designs always attempt to minimize power consumption by using the lowest clock speed that is necessary for the task at hand, then keeping the device in sleep mode as long as possible. This dual strategy pays out in devices that a.) are waken up occasionally by an external stimulus or b.) are waken up periodically but are sleeping for protracted times.

On the other hand, control algorithms must be engaged continuously and without the benefit of long-lasting power-down modes. Control strategies, such as PPF, proportional-integral-derivative (PID) or linear quadratic (LQ) consume relatively a steady computational effort within the allocated sample time. The ratio of active and idle times is much less than e.g. wireless sensor systems, and may be measured at most by tens; instead of thousands.

The goal of waking the microcontroller from sleep mode presents another issue for digital feedback control applications. While plants with slow dynamic response—such as thermal control—require sampling in the order of seconds, vibration control systems necessitate sub-second sampling periods. Waking an MCU using a real-time clock (RTC) is then not possible, as these peripherals do not keep time with a sub-second precision. Watchdog timers produce an unstable signal reliant on internal oscillators and, yet again, are not suitable to manage sampling periods in the sub-second domain. The remaining possibilities of waking the controller for each sample highly depend on the given processor architecture, but in general, we are left with an interrupt-based solution utilizing an external clock signal and keeping asynchronous timers awake in idle modes.

Given these considerations, the next sections shall compare the power savings that are possible to achieve by throttling clock speeds and by sending the microcontroller to sleep at idle times. The hope is to find a definite answer as to which of these approaches are better suited for digital control systems with sub-second sampling, such as, typically encountered in active vibration control.

2.1. Evaluating preferable current saving strategies

The reduced average current consumption I_r [mA] of the microcontroller caused by alternating active and sleep states may be approximated by

$$I_\mathrm{r} = I_\mathrm{p} \frac{t}{T}, \tag{1}$$

where I_p [mA] is the peak active current, and t/T [-] is the duty cycle of the active vs. inactive state of the microcontroller. The reader may note that, although it is low, the power consumption in sleep mode is not zero. This current is then usually included in the approximation (e.g. [3, 17]), yet we chose to neglect it in Eq. (1) because the microcontroller remains in this state only for short times, instead of the extended sleep states used in certain devices, such as low power wireless sensors.

On the other hand, reduced microcontroller current achieved by throttling clock speed may be

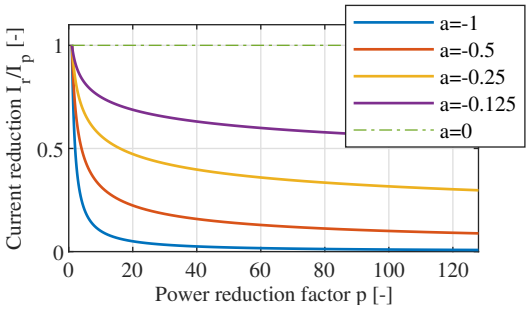


Figure 2: Dependence of current consumption on speed modelled by a power function.

approximated by the power function

$$I_\mathrm{r} = I_\mathrm{p} p^a, \tag{2}$$

where p [-] is a power reduction factor given by $p = 2^s$ with a slow-down level of $s = 0, 1, 2, \ldots$ that is natively encoded in binary and where $a < 0$ is an approximation parameter depending on the given microcontroller. The parameter $a = 0$ would mean that speed reduction provides no current consumption reduction, while $a = -1$ would mean that for each bisection of the speed the base peak current is also halved, making $-1 < a < 0$ a reasonable assumption for most microcontrollers (see Fig. 2.) Note that although the relationship of the current and the actual clock frequency is linear (cf. [3]), approximating the throttling factor by a power function instead enables us to generalize later discussion by cancelling the peak current.

Let $D = T/t$, then the MCU can be slowed down exactly D times before we hit the hard real-time sampling constraint and thus $D > 1$. However, clock speed prescalers—as everything else in digital computers—are encoded in binary, thus we must search for the base-two logarithm of this ratio, which can only be approximated by the nearest prescaler below our result, e.g. the maximal slowing factor is

$$p_\mathrm{max} = 2^{\lfloor \log_2 D \rfloor}, \tag{3}$$

where $\lfloor x \rfloor$ denotes the floor operation on a real number x, thus effectively discarding decimal places. Deciding which power saving strategy is used may be reduced to comparing their average current consumption. If the ratio of the current with reduced clocking and the sleeping strategy is larger than one, it is better to use the sleeping strategy, which we may express by

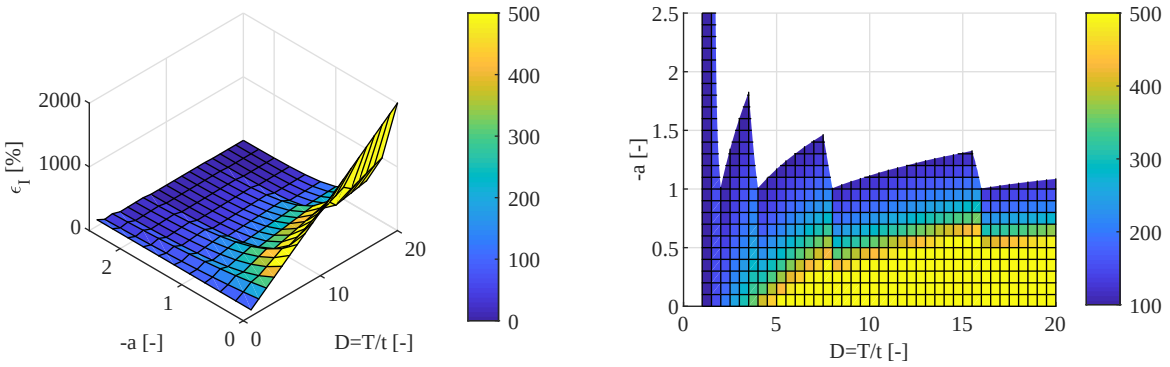
$$\frac{I_\mathrm{p}(2^{\lfloor \log_2 D \rfloor})^a}{\frac{I_\mathrm{p}}{D}} > 1, \tag{4}$$

obtaining the expression

$$D(2^{\lfloor \log_2 D \rfloor})^a > 1 \tag{5}$$

after simplification. From this it follows that the percentual efficiency gain ϵ_I [%] is then $\epsilon_\mathrm{D} = (2^{\lfloor \log_2 D \rfloor})^a \cdot 100$.

Plotting the function of power saving efficiency ϵ versus the ratio of available idle time D and the power reduction factor a will help to understand the dependency better. This is illustrated in Fig. 3, where Fig. 3(b) shows the edge case for this decision. As we can see for all active/idle ratios D and up to $a = -1$ the power toggling strategy pays greater dividends in current saving. It is unlikely that power will fall at a greater rate, thus we can conclude that sleeping modes are always to be preferred when implementing a battery-operated device running a feedback control application.



(a) Current saving factor plot.

(b) Current saving strategy decision boundary.

Figure 3: Current saving ratio ϵ_I depending on the active/idle ratio D and the power reduction factor a.

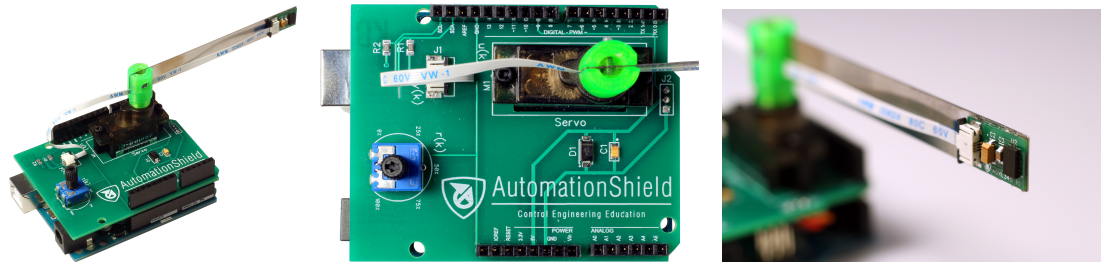
3. CASE STUDY: IMPLEMENTATION AND VERIFICATION

3.1. Hardware description

The proposed current sampling framework and the power-saving decision strategy has been verified using the miniature flexible link laboratory device illustrated in Fig. 4. As lighter structures and faster manipulation speeds become more common, researchers must study and control the vibration response of robotic arms, in addition to their kinematic properties. A flexible rotational link device then allows one to test vibration estimation and control strategies in a controlled laboratory environment. The open-source laboratory device called "LinkShield"[5] considered in this study contains a micro-servo motor as an actuator, to which a thin flexible beam is mounted. The dynamic response of the beam is monitored by an acceleration sensor placed at the tip. The reader may refer to [16] for more details on this experimental hardware.

The hardware is connected to an Arduino Uno microcontroller prototyping board that features the AVR-architecture Microchip Technology ATmega328P MCU. This board has been slightly modified for accurate current consumption reading. The LED reporting the power state of the board has been removed, and the power supply trace leading to the processor has been broken to allow for the insertion of a current probe. Static current readings have been measured by an AIM TTi 1908 benchtop multimeter, while dynamic current and voltage readings have been gathered by a CMikrotek µCP100 µCurrent Probe connected to a Digilent Analog Discovery 2 PC-based oscilloscope and data

[5] *see* the open-source and non-commercial AutomationShield project at http://www.automationshield.com



(a) Miniature flexible link device affixed to an Arduino Uno.

(b) Top view showing components.

(c) Beam tip detail with the accelerometer module.

Figure 4: Miniature low-cost flexible link device.

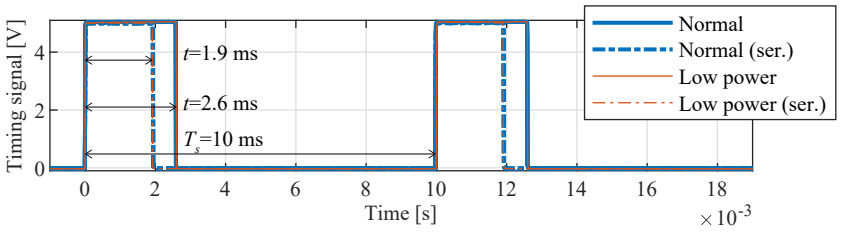


Figure 5: Active algorithm states are denoted by a 5 V signal, while inactive by 0 V. The total execution time t [s] of the algorithm is signified by the width of the pulses.

logger. The external clock signal representing sampling periods was generated using an Agilent 33521A benchtop function generator.

3.2. Control algorithm and its timing

Since the dynamic response of the beam is dominated by its first resonant frequency, the control algorithm may be effectively reduced to a simple PPF controller. A sampling period of $T = 10$ ms (100 Hz) is a reasonable and typical choice for such a system especially, given that the beam and a small tip mass result in the first resonance at 16 Hz.

The controller is implemented using the AutomationShield API [6] using the Arduino IDE. The application programming interface (API) takes care of the servo functionality using the default Arduino servo library and reads the acceleration data from the micro-electromechanical system (MEMS) accelerometer.

Since the first resonant frequency practically dominates the entire response of the system, we can safely assume that the relationship between the angular input position of the servo motor $u(t)$ and the position of the beam tip $q(t)$ can be expressed by the transfer function

$$P(s) = \frac{Q(s)}{U(s)} = \frac{c\omega^2}{s^2 + 2\zeta\omega s + \omega^2}, \tag{6}$$

where, after identifying the parameters based on measurements, we obtain an natural angular frequency of ω=100.4 rad·s^{-1} (16 Hz), damping ratio of $\zeta = 0.0027$ [-] and the actuator constant of c =-9.272E-4 m·deg^{-1} [16].

To test the energy-efficient sampling framework, we have considered the positive position feedback controller [7, 13], and modified it slightly to account for the acceleration reading to get [16]

$$G(s) = \frac{U(s)}{Q(s)} = -g\,\omega_c^2\,\frac{1}{s^2 + 2\zeta_c\omega_c + \omega_c^2}. \tag{7}$$

Choosing $\omega_c = \omega = 100.4$ rad·s^{-1}, $\zeta_c = 0.04$ [-] and the gain $g = 2$ deg·m^{-1} this continuous transfer function will be translated to the discrete-time with the choice of our sampling. Interested readers may consult our previous work in [16] on more details about the modelling, control and construction of the experimental device.

We must now time the total execution time of the control algorithm, including all the auxiliary duties. Figure 5 shows the hardware-measured execution time of the PPF application running on the MCU. The traditional implementation (blue) requires 2.6 ms (D=3.8 [-]) including serial data logging functionality, and 1.9 s (D=5.3 [-]) for basic use. The timing details of the current-saving sampling framework (red) are also foreshadowed here; it is obvious that the timing profiles of the two strategies do not differ.

3.3. Implementing low power modes

An extension to the AutomationShield API has been created to implement the low-power functionality of the AVR architecture for the purposes of this study. This extension is contained in the `LowPower.h` header and `LowPower.cpp` implementation file, automatically creating an instance of the `LowPower` object. When calling

```
LowPower.begin();
```

the general purpose input-output (GPIO) pins are set to output mode to save current in active mode, as per manufacturer recommendations [12]. The power-down sequence of the microcontroller is then launched by

```
LowPower.powerDown();
```

which disables the ADC, initializes the power down mode, disables BOD and finally executes the `sleep` command that is implemented in inline assembly.

The microcontroller can be then woken up from its power down mode by a number of methods; a reset signal, watchdog-timer (WDT) overflow or detecting an edge on an external signal connected to a GPIO pin. This particular architecture may be sent to a higher-tier sleep mode, from which it can be woken by an asynchronous timer as well. Implementing sampling for control and estimation for vibration systems typically requires a rate beginning from tens of Hz up to the kHz range. Other than the fact that the WDT is imprecise as its tied to an internal oscillator, it can only generate interrupts upon overflow but it lacks a register match option. The WDT is thus suitable for slow sampled applications but not for faster rates such as needed for AVC. The asynchronous timer could be a good option, unfortunately the crystal pins on the 328p are shared with the main clock of the MCU, requiring the internal 8 MHz oscillator to be engaged for the main clock, thus reducing the speed of computations by default.

This particular MCU is then left with a single option: using an external tick, coming in the form of a full-swing logical oscillator. Such a signal can be supplied by a serial port programmable oscillator generating a square signal, such as the Linear Technology LTC6904/LTC6904 [11]. We have used a signal generator to initiate the interrupt signal, as a discrete oscillator chip was unavailable at the time of conducting our experiments. Waking the MCU from sleep then requires an interrupt associated with a compatible pin in the application.

Finally, the default clock speed—in this case 16 MHz—can be slowed down a factor of p by calling

```
LowPower.slowDown(p);
```

and where $p = 2^s$ with $s = 0, 1, \ldots, 7$.

3.4. Static current consumption

The AVR architecture and ATmega328P provides several power saving measures. Of these we are primarily interested in reducing power in power-down mode and the relationship of current and throttled clock speed. First, let us look at the current consumption of the MCU in active and power down modes. Table 1 lists power measurements of the ubiquitous blinking LED example. Changing the impedance state of the general purpose input and output pins saves about 3 mA, while disabling the analog to digital converter and the brownout detection peripheral advances current consumption into the nanoampere range.

Figure 6 shows the measured microcontroller current in dependence of the power reduction factor. The blue graph on the top denotes baseline naive implementation, while the bottom red illustrates the case when GPIO ports are switched to output. Black crosses denote datasheet figures given by the manufacturer [12]. Measurements are shown in dash-dot lines, while approximations by Eq. (2) using full lines.

Table 1: Current consumption of the ATmega328P microcontroller in various modes.

State	Current	
Active (LED on)	15.8	mA
Active (LED off)	15.3	mA
GPIO as outputs	12.0	mA
Power down	144	µA
ADC disabled	23	µA
BOD disabled	<0.2	µA

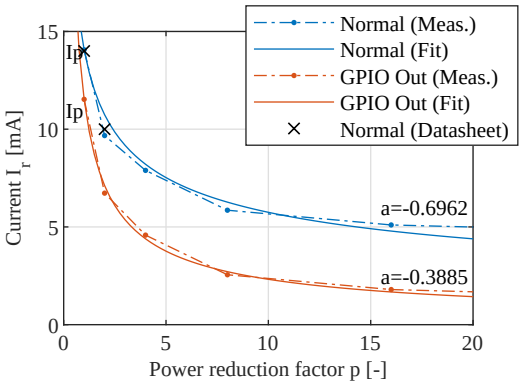


Figure 6: Baseline naive current consumption (blue) and disabled GPIO pin implementation (red) for PPF-based active vibration control. Dash-dot lines denote measurements, full lines approximations.

3.5. Current consumption prediction and verification

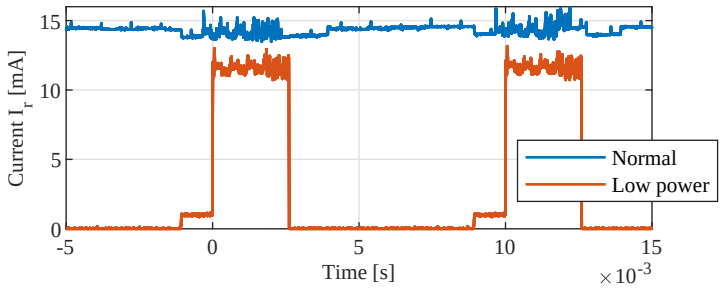
A naive implementation would consume approximately 14 mA. The maximal possible rate of throttling clock speed with serial output enabled according to Eq. (3) corresponds to a factor of $p_{\max} = 2$ (and $p_{\max} = 4$ without serial communication), assuming the current algorithm with TET t and sampling T. Let us assume that GPIO is set such that power is saved in active mode, that is peaking at 11.5 mA. Throttling clock speed results in a predicted reduced current of $I_r = 7.1$ mA (4.4 mA w/o serial) according to Eq. (1) and a measured real consumption of $I_r = 6.7$ mA (4.6 mA w/o serial). The decision in Eq. 5 predicts a 237 % gain in power savings when using the current-saving sampling framework, consuming 3.0 mA with serial communication included.

Let us verify this in experiment. Figure 7(a) shows the average current consumption of the microcontroller with the current-saving sampling framework at 3.0 mA, as predicted by our calculations. Although the on-board regulator keeps the voltage level fairly constant, we may obtain a more precise power consumption analysis by a separate voltage signal and calculating $P_k = U_k I_k$, where U and I are instantaneous voltage and current at time sample k. A naive implementation requires about 70 mW of power, a current-saving sampling framework reduces MCU energy needs to 15 mW (see Fig. 7(b)).

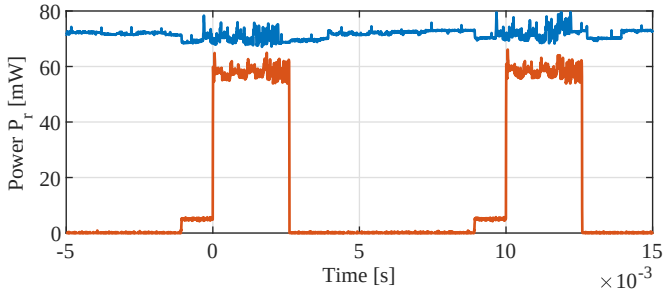
4. CONCLUSIONS

This paper compared power saving strategies for feedback control applications with sub-second sampling in theory and experiment. Our results suggest, that operating the microcontroller at full clock speed, then sending it to a low power mode is likely to be preferred in all possible scenarios and microcontroller architectures. We have verified our results by testing this on the PPF algorithm, where the low-power sampling framework reduced microcontroller power expended to run the control algorithm by a factor of five, not accounting for the power consumption of sensors and actuators.

Even though the sensors are sampled at short intervals only, for certain actuators the input signal must be maintained throughout the entire duration of the sample. The servo motor utilized in our study



(a) Current (I_{avg}=14.3 mA vs. $I_{\text{r,avg}}$=3.0 mA).



(b) Power (P_{avg}=71.4 mW vs. $P_{\text{r,avg}}$=15.2 mW).

Figure 7: Current and power consumption of the PPF algorithm using a baseline implementation and the current-saving sampling framework.

is a good example for this, as it must receive a pulse-width modulation signal to maintain its prescribed position. As the AVR architecture considered in our verification example turns off the hardware timer required to maintain the servo position in LPM mode, the properties of the feedback loop are considerably affected. Circuit designers and embedded programmers thus must pay close attention to this limitation, and choose a microcontroller architecture that enables LPM and maintaining actuator signals at the same time.

We have not included the choice of slowing down the processor as much as possible, then sending it to sleep in our study. This power saving method could form a basis of further work, where the wake-up time and wake-up current of the microcontroller shall be also included in the analysis. Whether such a strategy is more efficient than sending the MCU to sleep after a full-speed cycle remains to be revealed.

There are microcontroller architectures that feature a wider feature set for LPM than the AVR chip considered in this paper. More advanced MCU designs may broaden the capabilities or ultimate efficiency of the low-power sampling framework presented in our discussion. Waking up the microcontroller from sleep is another factor that requires more attention, as we have not included the power requirements of the external oscillator in our present study. An external programmable square-wave oscillator requires 1.7 mA additional current, while a hardware configurable component can be run on 500 µA — ultimately increasing the overall energy requirement, but does not invalidate out general conclusion. Better wake-up sources suitable for the typical range of sub-second sampling must thus be evaluated in upcoming studies.

Further work shall test the sampling system on a wider range of microcontrollers and architectures and shall investigate the possibility to create an energy-saving sampling system for optimal control algorithms, such as model predictive control.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the contribution of the Slovak Research and Development Agency (APVV) under the contracts APVV-18-0023, APVV-17-0214 and APVV-14-0399. The authors appreciate the financial support provided by the Cultural and Educational Grant Agency (KEGA) of the Ministry of Education of Slovak Republic under the contract 005STU-4/2018.

6. REFERENCES

[1] M. M. Al-Kofahi, M. Y. Al-Shorman, and O. M. Al-Kofahi. Toward energy efficient microcontrollers and Internet-of-Things systems. *Computers & Electrical Engineering*, 79:106457, 2019.

[2] A. Altan and R. Hacıoğlu. Model predictive control of three-axis gimbal system mounted on UAV for real-time target tracking under external disturbances. *Mechanical Systems and Signal Processing*, 138:106548, 2020.

[3] A. Cebrian, J. Rey, A. Tormos, and J. Millet. Adapting power consumption to performance requirements in a MSP430 microcontroller. In *2005 Spanish Conference on Electron Devices*, pages 83–86, Feb 2005.

[4] C. Chakrabarti and D. Gaitonde. Instruction level power model of microcontrollers. In *1999 IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 1, pages 76–79, May 1999.

[5] T. Doughty, J. Heintz, and M. Ishii. Reducing Parkinsonian hand tremor with a novel dynamic eating utensil. volume 3B: Biomedical and Biotechnology Engineering of *ASME International Mechanical Engineering Congress and Exposition*, 11 2013. V03BT03A012.

[6] G. Takács *et al*. AutomationShield: Control Systems Engineering Education. Online, 2018-2020. [cited 17.1.2020]; Available from http://www.automationshield.com.

[7] C. J. Goh and T. K. Caughey. On the stability problem caused by finite actuator dynamics in the collocated control of large space structures. *International Journal of Control*, 41(3):787–802, 1985.

[8] M. Hiraki, K. Fukui, and T. Ito. A low-power microcontroller having a 0.5μa standby current on-chip regulator with dual-reference scheme. *IEEE Journal of Solid-State Circuits*, 39(4):661–666, April 2004.

[9] Insik Shin and Insup Lee. Periodic resource model for compositional real-time guarantees. In *RTSS 2003. 24th IEEE Real-Time Systems Symposium, 2003*, pages 2–13, Dec 2003.

[10] D. Lautner, X. Hua, S. DeBates, M. Song, and S. Ren. Power efficient scheduling algorithms for real-time tasks on multi-mode microcontrollers. *Procedia Computer Science*, 130:557 – 566, 2018. The 9th International Conference on Ambient Systems, Networks and Technologies (ANT 2018) / The 8th International Conference on Sustainable Energy Information Technology (SEIT-2018) / Affiliated Workshops.

[11] Linear Technology Corporation. LTC6903/LTC6904: 1 kHz to 68 MHz serial port programmable oscillator. Online. Cited 11.2.2020. Available at https://www.analog.com/media/en/technical-documentation/data-sheets/69034fe.pdf, 2003. Datasheet, Rev.: LT 0312 REV E.

[12] Microchip Technology and Atmel Corporation. ATmega328P: 8-bit AVR microcontroller with 32k bytes in-system programmable flash. Online. Cited 11.2.2020. Available at http://ww1.microchip.com/downloads/en/DeviceDoc/

Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf, 2015. Datasheet, Rev.: 7810D-AVR-01/15.

[13] A. Preumont. *Vibration Control of Active Structures*. Kluwer Academic Publishers, 2. edition, 2002.

[14] F. Reverter and M. Gasulla. Experimental characterization of the energy consumption of adc embedded into microcontrollers operating in low power. In *2019 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pages 1–5, May 2019.

[15] Y. Shenq, W. Sheik, A. Mazlan, and Z. Ripin. Development of add-on active vibration control (AVC) for power tools. In *Innovate Malaysia Design Competition 2015*, Kuala Lumpur, Malaysia, 8 2015.

[16] G. Takács, M. Vríčan, E. Mikuláš, and M. Gulan. An early hardware prototype of a miniature low-cost flexible link experiment. In *27th International Congress on Sound and Vibration ICSV, Prague, Czech Republic*, pages 1–8, July 2021. Accepted.

[17] X. Zuo and Y. Liu. Low power performance achievement in embedded system. In *2011 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC)*, pages 4655–4658, Aug 2011.